EPiC Series in Computing

Volume 55, 2018, Pages 148–161

GCAI-2018. 4th Global Conference on Artificial Intelligence

Classifier Labels as Language Grounding for Explanations

Sai P. Selvaraj[1], Manuela Veloso[2], and Stephanie Rosenthal[3]

[1] spandise@alumni.cmu.edu, Carnegie Mellon University, Pittsburgh, USA
[2] mmv@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, USA
[3] s.rosenthal@chatham.edu, Chatham University, Pittsburgh, USA

Abstract

Advances in state-of-the-art techniques including convolutional neural networks (CNNs) have led to improved perception in autonomous robots. However, these new techniques make a robot's decision-making process obscure even for the experts. Our goal is to automatically generate natural language explanations of a robot's perception-based inferences in order to help people understand what features contribute to these classification predictions. Generating natural language explanations is particularly challenging for perception and other high-dimension classification tasks because 1) we lack a mapping from features to language and 2) there are a large number of features which could be explained. We present a novel approach to generating explanations, which first finds the important features that most affect the classification prediction and then utilizes a secondary detector which can identify and label multiple parts of the features, to label only those important features. Those labels serve as the natural language groundings that we use in our explanations. We demonstrate our explanation algorithm's ability on the floor identification classifier of our mobile service robot.

1 Introduction

Service robots can autonomously generate and execute plans to perform tasks for humans while handling the uncertainty of their surroundings. In order to achieve these state-of-the-art results, many service robots have adopted deep learning techniques, especially for perception using Convolutional Neural Networks (CNNs). While these algorithms help robots achieve better performance in real-world settings, they are challenging to understand or interpret by both experts and non-experts.

Because non-experts in the environment may need information about a robot's perception, state prediction, planning, and execution as it performs its tasks, our goal is to generate explanations of the robot's algorithms. Prior work in explaining robot behavior has focused on either explaining an action policy (e.g., [6]) or narrating a particular sequence of execution (e.g., [15]). While these explanations can potentially help people understand a robot's actions, they do not help a person understand how the robot determined its current state before performing those actions (e.g., using a classifier).

Explaining classifier predictions, especially CNN classifiers, is a growing area of research. One popular approach to generating explanations for a classifier is to learn an interpretable model is what usually people do but the idea is to learn interpretable models based on the predictions of the original model [14]. This technique has been very successful in generating domain specific explanations interpretable by the experts but it may still be hard for non-experts to understand the new local classifier. Other approaches explain perception classifiers visually by overlaying heatmaps on the original images to help people understand what parts of the image were important (e.g., [17, 20, 21, 24]). While a visual representation can be useful for many applications, there are times, especially when a robot is moving, when it cannot display an image for explanation. Additionally, a visual technique is challenging to extend beyond the visual domain.

In this work, we are interested in automatically generating natural language explanations rather than visual explanations or learning simplified models. While natural language may be more suitable for human understanding, it is also challenging to generate because there often is not a grounding that easily translates a classifier's features to language and it is unclear which features should be explained. To overcome these challenges for the task of explaining a bird species classifier, Hendricks et al. [8] uses a pre-defined natural language description of the classes to train a network to produce explanations. However, this process of pre-defining explanations is impractical when there are no descriptions for the classes, when it is unclear which features contribute to the classification, or when there are many classes to explain.

We contribute a three-part algorithm for explaining a CNN classification that identifies a subset of features to explain and does not require a predefined mapping from features to natural language. In the first step, our algorithm identifies the important pixels in the image that contribute most to its classification using deep visualization techniques. Then, the algorithm labels only those important regions of the image using a secondary pre-trained off-the-shelf classifier. The labels serve as our natural language descriptions of the important features. Finally, the algorithm stitches together the labels and their locations within the image using templates for a final explanation.

We demonstrate our explanation algorithm on our robot's floor identification CNN classifier. After our robot captures an image in our building and classifies the floor it is on, it then identifies important features in the image that helped it determine its floor prediction, labels those important regions of the image using an off-the-shelf object detector, and uses those labels to produce a natural language explanation. We conclude with practical implementation results that would allow our algorithm to be applied to other vision-based domains as well as non-visual domains where important features used to achieve the classification result can be labeled.

2 Related Work

Many different techniques have been proposed to study how machine learning algorithms make predictions and how robots use those predictions to make action decisions. We distinguish between two types of explanations - those that explain a specific observation or feature vector and those that explain the overall behavior.

First, many algorithms have been proposed in order to explain or understand the overall behavior of a machine learning algorithm and in particular understand which features are most important in their classification decisions. Finding the important features or performing feature reduction in the context of machine learning is a well-researched area [3, 4]. More recently, a set of approaches for generating local explanations for a particular feature subspace aim to be more interpretable by a machine learning expert than the original models that they are generated from [2, 14, 16]. In Ribeiro et al. [14], the authors generate the explanations in the corresponding domain for the classification of any model by learning a separate sparse linear model that locally approximates the original model to explain its decisions.

While these algorithms have all been very successful, we are interested in understanding why particular decisions or predictions are made, and not the general rules for the classifier. For example in image classification, an algorithm could find a table and chairs and a potted plant as important for determining that the image was taken on the hallway of a building. However, in the particular image, the potted plant was missing but it was still classified as the hallway. In this work, we are interested in knowing the particular features of this image (e.g., the table and chairs) that contribute to classification rather than the general features for the entire class (e.g., table, chairs, and plant).

Image-specific visualization techniques determine what features CNNs have identified as important in each image [17, 18, 20, 21, 24]. Essentially, they serve as functions that find important regions in an image that most affect the classification and display a heat map representing the relative importance of each pixel over the original image. A variety of different methods have been proposed for determining which features are important, leading to different explanations of the classification predictions of CNNs. Zeiler and Fergus [20] uses an occluding patch to systematically occlude parts of the input image, and use the classifier's confidence on these images to generate a heat map. The idea behind the approach is if a key feature in an image gets occluded, then the classifier's confidence will fall upon the occlusion. Gradient visualization technique [18] uses gradients of the classifier's score with respect to the pixels in the input image as the heat map. The rationale is that the probability scores are more sensitive to the change in values of the important features than others.

Several challenges prevent these visualization techniques from being applied to other domains. First, there are times when it is not feasible to display a heat map on a robot for an explanation or the heatmap can not explain the classification. Second, there are many other types of algorithms for non-visual features including neural networks that could also be explained by finding important features, but the visualization as heat map does not make sense for non-visual features. In this work, we focus on producing natural language explanations of high-dimension data including images.

Hendricks et al. [8] focus on generating textual explanations for fine-grained classification of 200 bird species using a deep network from their images and uses manually-generated bird descriptions as the training data. The algorithm explains the ResNet He et al. [7] features extracted from the entire image, conditioned on both the image and the class predictions. While such an approach might be possible for cases where the vocabulary for explanations are easy to manually define or pre-existing in some other form, it is very tedious to generate natural language for new domains. Similarly, the survey paper on explanations and explainable systems by Abdul et al. [1] revealed that works in the space of explainable AI focus on mathematics to create a mathematically interpretable model to explain a complex model but usually do not consider how non-experts can interpret the explanations. They would manual mapping between features and natural language to be usable, which is neither feasible nor practical.

Explanations of robot states and actions also have similar challenges in mapping features or states to natural language. For explaining robot actions, Hayes and Shah [6] aims to synthesize policy descriptions and respond to queries about why robots do or do not perform particular actions in particular states. They learn a simplified domain model of the environment from demonstrations then use statistics of the planner over the data to create a behavioral model. However, their natural language explanations require manually defined predicates and corresponding language. Rosenthal et al. [15] creates narrations of robot paths and also requires predefined dictionaries mapping states to language. In this work, we will utilize existing off-the-shelf classifiers to label important features of each particular image with natural language descriptions of the pixel regions, eliminating the need for manual labeling.

3 Explanation Problem and Approach

In this work, we assume that a pre-trained classifier C and an observation (e.g., an image) I with possibly many features are given to an explanation algorithm. Our goal is to automatically generate a natural language explanation of why C classified I as a given class $y \in Y$.

In order to accomplish this task, we must overcome two challenges. First, unlike a visual explanation that can produce a heat map overall features, linguistically explaining each of many features is infeasible and not necessarily interpretable by people. Thus, we must determine the most important features to explain the classification. The other challenge is that there may not be a mapping from those important features to natural language to use in our explanation. In order to overcome this challenge, we make use of off-the-shelf classifiers that can label parts of the feature space. In the case of image-based explanations, we use a multi-object classifier that labels objects within images. Our resulting explanation algorithm is comprised of the following main steps:

1. Identify important features for the classification

2. Label the important features, and

3. Generate an explanation using feature labels as natural language groundings.

We describe our algorithm in the context of explaining the classification of images for a mobile service robot application, though the algorithms described are potentially applicable not only to the images but also to any other classifier where there are algorithms to find important features and to label subsets of those features.

Step 1: Identifying Important Regions

We assume that a classifier C outputs $p(I = y|w)$, the probability of an image $I \in [0,1]^{a*N}$ with a channels (i.e., 3 for R,G,B) and N pixels having classification $y \in Y$ given the trained parameters w. For clarity, we will refer to the ith pixel in the image as $I[i]$. Our first step is to find the important pixels that contribute most to the classification of I as y.

Importance Functions

Given C and I, an importance function `importance`(I, C), ranks the pixels in the image based on their impact on the classification of the image as y. The importance function outputs a heat map $H \in [0,1]^N$ that contains a measure of relevance of each pixel $I[i]$ to the class y. The heat map will have higher values for those pixels that are considered important, and lower values otherwise. For example, Figure 1(b) shows the visualization of the heat map for image in Figure 1(a). A variety of importance functions, each with their own heat map, have been proposed for explaining the classification predictions of CNNs [18, 20, 21].

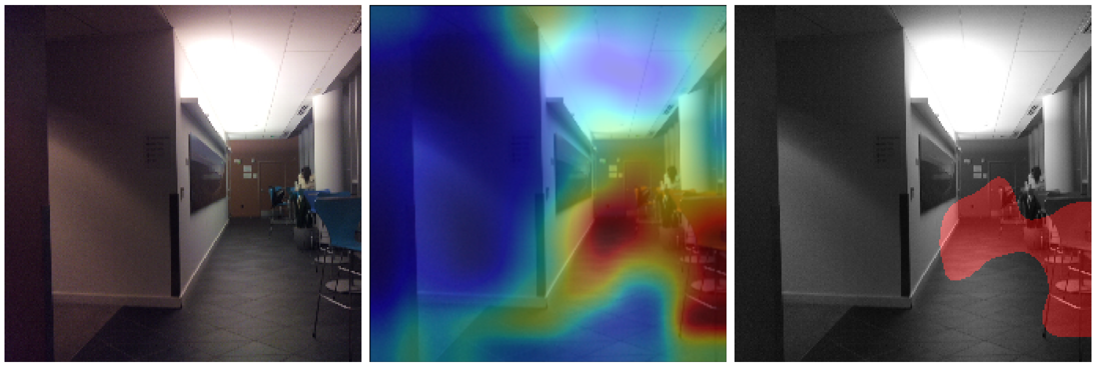

(a) (b) (c)

Figure 1: The importance function takes an original image (a) and produces a heat map (b) representing the importance of each pixel, where red and blue represents the most and the least important pixels respectively. Our algorithm thresholds the heat map to produce a binary mask (c) for top– based on importance, ρ=5% of pixels.

Gradient Visualization Technique

In this work, we use the Gradient Visualization Technique [18] for its simplicity. For this technique, H represents the magnitude m of the derivative of the classification confidence with respect to the image. The magnitude of ith pixel m_i represents the sensitivity of the network's prediction to the change in that pixel's value and is equal to the derivative of the classification probability $p(I = y|w)$ with respect to $I[i]$. We expect the classifier accuracy to be more sensitive to the change in values of the important features than that of non-important features. Note that since the gradients are pixel-wise importance values for the image, the heat map is generally of high entropy and thus lacks continuous important image regions.

Importance Mask

While the heat map is useful for visualization, we propose the use of a binary mask to signify whether a feature (pixel) is included in the important region or not. Concretely, a binary mask $M \in \{0,1\}^N$ is created such that each pixel i takes value:

$$M[i] = \begin{cases} 1 & \text{if important} \\ 0 & \text{otherwise.} \end{cases}$$

The red mask in Figure 1(c) is an example of a binary mask.

To segment the heat map into important and not important regions, several different techniques have been proposed – region-based [5], threshold-based [9] and model-based [11]. Although model-based segmentation tends to perform better, for simplicity, we use a simple threshold-based segmentation to find the important regions in this work. The threshold-based approach segments the top ρ% of the pixels from H. For example, Figure 1(c) is the importance mask obtained by thresholding the heat map shown in Figure 1(b) with $\rho = 5\%$ of pixels. We note that any of the segmentation techniques could be used.

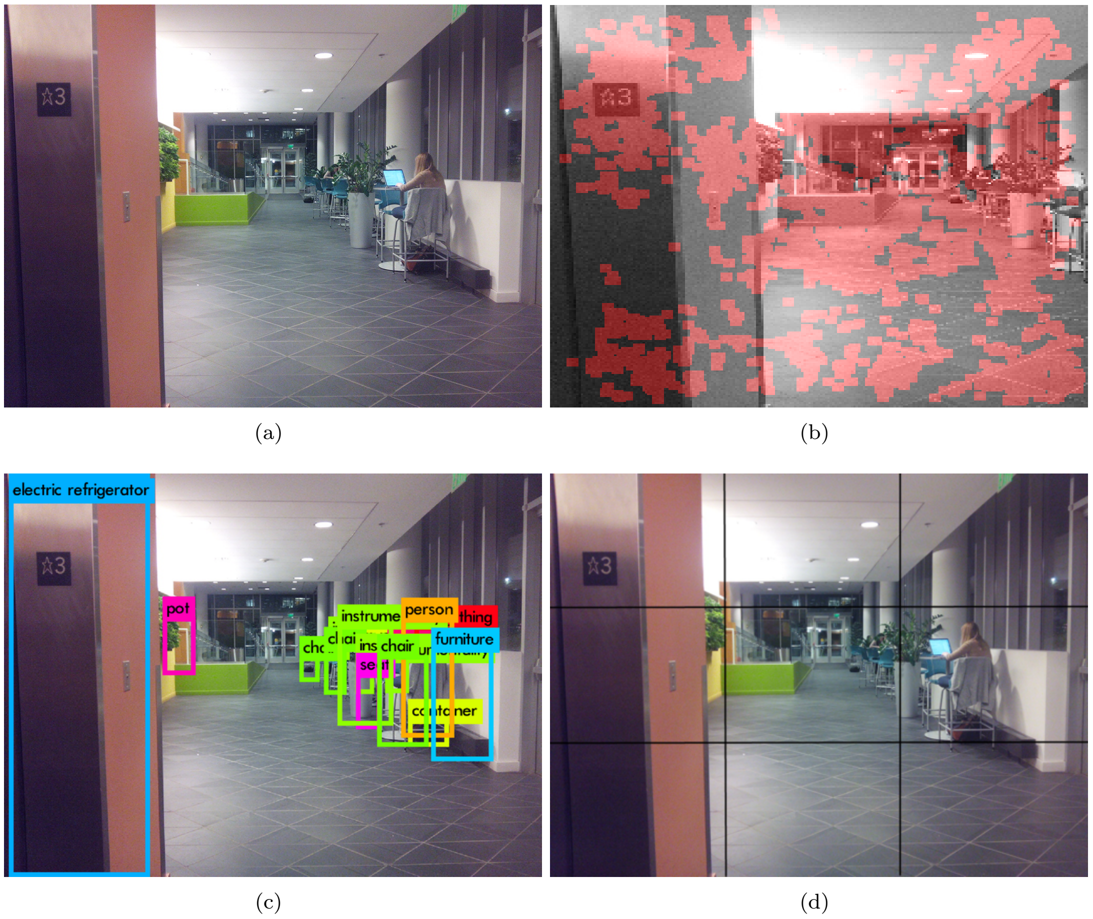


Figure 2: Example for generating E_M. Our algorithm takes an image (a) and creates a binary mask of the important features with ρ=50% (b). It also uses a secondary classifier to label regions of the image (c) and discretizes the image into 9 grid cells (d) to use in the explanation.

Step 2: Labeling Important Regions using a Secondary Classifier

Using the mask M of important features in I (e.g., Figures 1(a) and 1(c) and also Figures 2(a) and 2(b), our algorithm then generates natural language groundings for the important regions in the image masked by M using a secondary classifier. In our application, we use an image-based multi-object detector, but a different secondary classifier over features in I would work in other domains.

Extracting Explainable Features of an Image

The secondary classifier identifies a set of objects as potential explainable features E $\{o_0, ...o_l\}$ in the image and their corresponding bounding boxes $\{(xa_0, xb_0, ya_0, yb_0),\ ...\ (xa_l, xb_l, ya_l, yb_l)\}$ as shown in Figure 2(c). The bounding boxes represent the features that pertain to the object.

To generate explainable features for the important regions in the image E_M, the algorithm removes the objects in E with less than a threshold ratio t, of their bounding box overlapping with the importance mask M. The algorithm also filter unrelated features in E by removing tuples containing objects present in a rejection corpus R. The filtering is done to remove erroneous detections from the multi-object detector. The rejection corpus is constructed using object names that would not be found in the robot's environment, for example, robot operating in building corridors will not find objects like 'deep-freeze' and 'car' which can be included in the rejection corpus.

To summarize, our algorithm automatically generates a list of explainable features E_M for a particular image I and importance mask M by applying filters:

$$E_b = \{e_i \mid \frac{sum(M[xa_i : xb_i, ya_i : yb_i])}{(xa_i - xb_i) * (ya_i - yb_i)} > t, \ \forall e_i \in E\}$$

$$E_M = E_b \setminus \{e_i \mid o_i \ \in R, \ \forall e_i \in E\}$$

where M[x1:x2, y1:y2] is a submatrix of M with (y2-y1) rows and (x2-x1) columns, containing elements from M in x1 to x2 columns and y1 to y2 rows.

Additionally, to simplify the references to the explainable features (in our case references are locations in the image), we discretize the image into g grid cells (e.g., Figure 2(d) with 9 cells), and use the grid index gId of the detected object's center as a proxy to its location in the image rather than the xy location. The discretization can be applied to non-vision data as well. For example, rather than using full timestamps in temporal data, it is possible to discretize events by morning, afternoon, evening, and night. In the end, our explainable features are a set of labeled objects that appear in the most important regions and their corresponding relative location within the image.

Extracting Explainable Attributes of a Class

At this point, our algorithm has generated a list of many explainable features for an image. In the next step, we would like to identify the explainable features that occur commonly in the class and that could be described as an attribute of the class rather than anomalous objects. Given a training set of images prior to providing any explanations, our algorithm computes the explainable features, the same way as discussed above, for each training image, and then computes the union (probabilistic combination can also be used) of E_Ms as the explainable attributes for class $y \in Y$ as E_y.

Note that we use a small training set of 5 so the union is still a small number of explainable class attributes. Other methods such as taking the intersection or applying a threshold may be used to reduce the size of the class attributes if necessary. We assume that the classifier, importance function, and multi-object detector are good enough to produce unique explainable features for each class, i.e., all the classes are distinguishable using only their explainable feature E_y. E_y represents the dictionary of attributes defining the class.

Step 3: Generating Explanations

Finally, our algorithm computes the intersection $E_{M,y}$ of features in E_M (representing the important regions of the image) and E_y (representing the attributes of class y) to find the explainable features that the two have in common. There may be features in the image that are not attributes of the class or vice versa that should not be explained. For example, if a plant is detected in a hallway but that plant is not commonly found there, it should not be reported as a reason that the image was classified as the hallway.

The algorithm use language grounding and template-based natural language generation to convert $E_{M,y}$ to natural language explanations. To ground the grid locations, the algorithm relates the grid location to a corresponding spatial location. For example, gId=(0,2) is 'top right' and gId=(1,1) is 'center.' For grounding explainable features, the multi-object detector's object labels are a proxy for naming the objects in the image. The algorithm then combines all the natural language groundings from $E_{M,y}$ with a simple natural language template to create an explanation. Our algorithm demonstration section includes the template used in our work.

Rather than explaining all pixels or specific pixels, the algorithm focuses on important pixels and finds natural labels only for those pixels, using a secondary classifier. It also focuses on explaining the important attributes of the class rather than all objects that could be labeled in the important regions. The resulting explanation is a summary of the objects or explainable features that are representative of the class.

4 Algorithm Demonstration

When a robot navigates across many floors of a building or multiple buildings, one major challenge that it has is localizing itself to determine which floor it is currently on. We collected a Building-Floor dataset and trained a CNN scene recognition classifier for determining which floor our robot is on in our building. We then tested our algorithm's ability to explain why it classified our robot's images as particular floors of the building. We used an off-the-shelf multi-object detector to label explainable features like chairs, plants, and tables in our hallways that distinguish one floor from another. Our results show that the robot's explanations are accurate yet concise given the number of features in the images.

Building-Floor Dataset

We collected a Building-Floor dataset in one of our buildings. Each image contains the scene just outside the elevator from six different floors of the building, ref Figure 3. The goal of the classifier C, trained on this dataset, is to find which floor the image is taken from. Since the robot cannot change floors other than by taking the elevator, the elevators are the only locations where we need to classify the robot's location.

For each of the floors in the building, five images were taken at a particular location that our robot stops at after exiting the elevator. To simplify the analysis, all the images were taken at the same time of the day, and the effects of people moving around in the building are not considered. The training data consists of three images, and the remaining two images form the verification dataset. As the scene does not change a lot in theory for the dataset that we chose, one image should be enough to train and obtain good performance. In practice, this is not true because of the large features space and the complexity in the image domain, so we use three images to avoid overfitting and capture some variation in the scene. Testing data collected separately consists of five images from each of the floor taken at similar daytime and settings as the training dataset. The newly collected testing dataset was obtained while our robot was performing multi-floor navigation tasks.

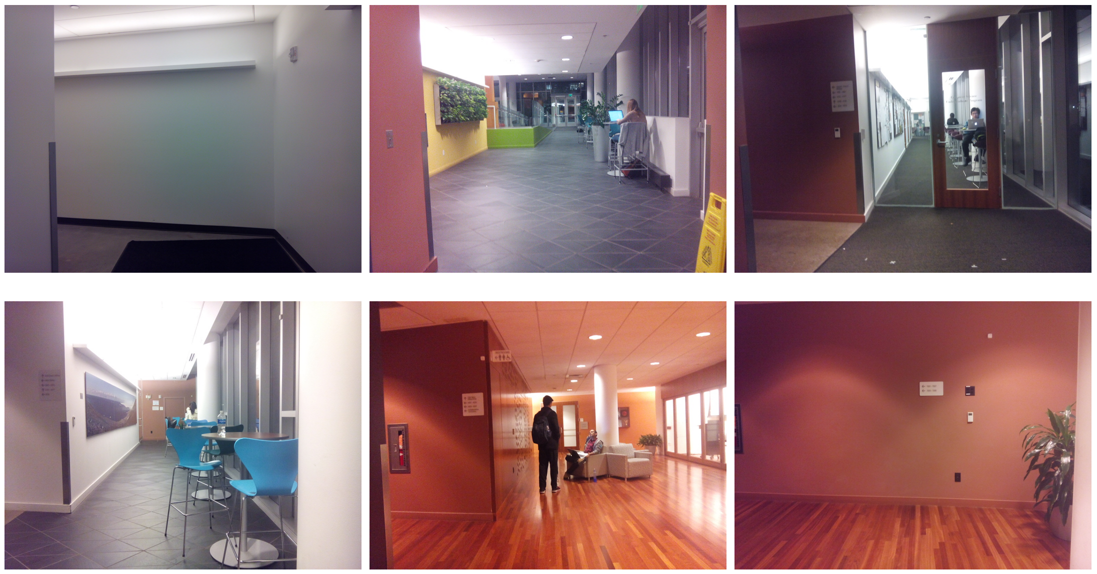


Figure 3: Sample of images from the Floor detection dataset. Each image belongs to a different floor.

Network Architecture

The deep learning network for our floor identification classifier C, consists of nine layers following AlexNet [10] in a modified Siamese architecture as proposed in Sun et al. [19], Zheng et al. [22], which combined the identification– Softmax, and the verification loss– Contrastive, for better performance, refer Figure 4. The approach we have taken to classify each of the floors using a CNN based network is currently a popular method and can be easily scaled to accommodate more classes. Our main reason for combining identification and verification loss with a pre-trained network is to reduce overfitting which could happen when the complexity of network is higher than that of the data. During training, the first seven layers of our network were initialized from Places205-AlexNet which was trained in the Places205-Standard dataset and provided by the authors [23]. The remaining two layers were trained from scratch. During training, the contrastive loss was utilized in the eighth layer which is a dense layer of 1000 units, while the softmax loss was employed in the ninth layer.

We found that the classifier C can classify all the images in the testing dataset correctly. We believe the high performance of our network is because the training and testing images were taken in similar setting and that the inter-class variation is quite high in the dataset. Another important reason for the good performance of our floor identification module is the use of the Siamese architecture during training. Siamese architecture by its nature increases the network's ability to maximize inter-class variability while minimizing intra-class variability.

Demonstration Setup

We implemented our explanation generation algorithm to describe the deep learning floor classifier. Given an image that was classified, the algorithm first finds the important regions of the image. Because the heat map generated from the gradient visualization technique is generally of high entropy and lacks continuous important image regions, we dilated the heat map twice with a 3x3 kernel. Dilating smoothens the heat map and improves the continuity of important

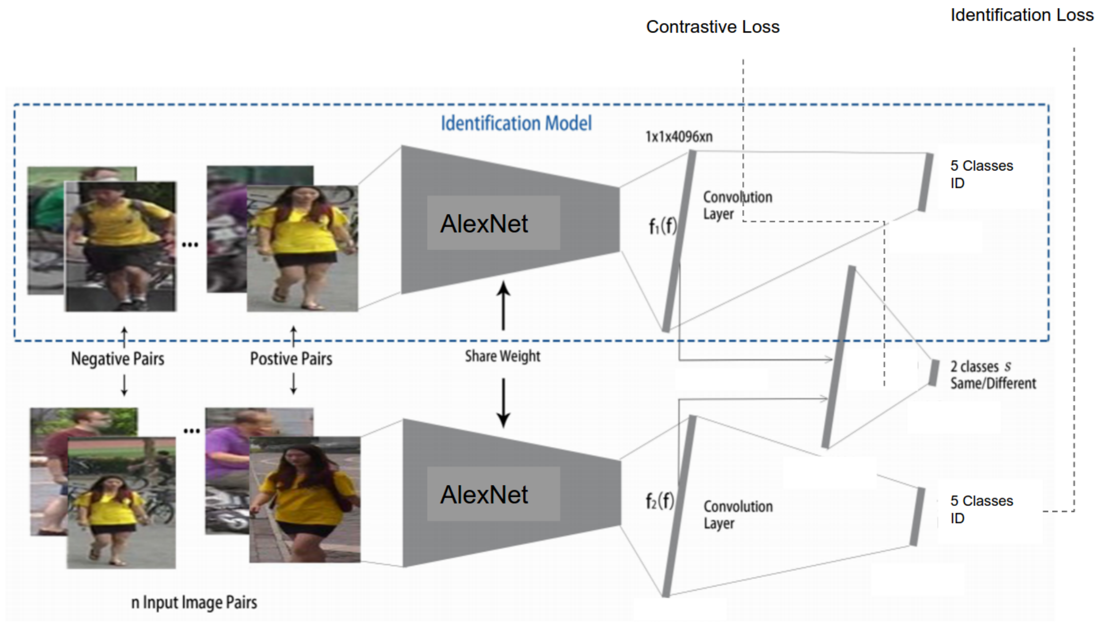


Figure 4: Architecture of modified Siamese network used for training the floor identification classifier

regions. The algorithm then generates the importance mask M, by segmenting top ρ=50% pixels from the dilated heat map.

In our demonstration, our algorithm used Yolo9000 [13] as the multi-object detector. The network weights were pre-trained and provided by the authors. Yolo9000 uses a hierarchical tree classification which is built using WordNet [12] concepts. For example, 'vehicle' would be above 'car' and 'bike'. For each image, Yolo9000 outputs a set of potential bounding boxes with a probability attached to each of the boxes and a label tree. We accept boxes with a probability greater than 0.1, and we set a threshold of 0.7 to determine which label in the tree to use (we prefer labels that are general and accurate to those that are specific but uncertain).

The hierarchical nature of the classification leads to some objects receiving labels near the top of the tree (more general labels). For example, some objects would be labeled 'instrumentality', 'things' or 'matter' rather than a more specific label. Our rejection corpus R is also constructed to remove these labels. For our building-floor domain, R = {'home appliance', 'living thing', 'container', 'artifact', 'person', 'conveyance', 'bottle', 'instrumentality', 'whole', 'deep-freeze', 'electric refrigerator', 'machine'}. We chose these labels based on our experiments on Yolo9000 using the domain knowledge. At least t = 0.5% of the pixels in the explainable feature bounding box must be in the importance mask for our algorithm to include it in E_M.

We use the same building-floor training and validation dataset to determine the explainable class attributes. The attributes are presented in the results section below. Additionally, our algorithm discretizes the image into g=9 grid squares. We defined groundings like 'top' for (0,1), 'right top' for (0,2), and 'center' for (1,1) gIds.

Putting all of the information together, our algorithm outputs natural language text based on our template:

I am in Location(y), because I see Label(O_0) at GridLabel(gId_0), Label(o_1) at

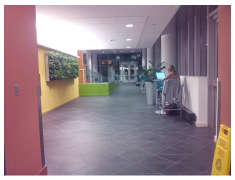
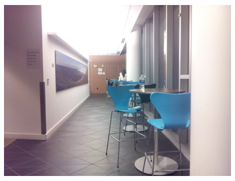
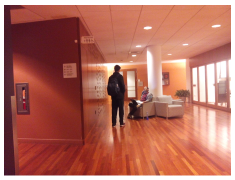
Image	Explanation algorithm output
	Classification: Floor 3 $E_{M,3}$: [‘(0,1), pot’, ‘(1,1), chair’, ‘(2,1), furnishing’, ‘(2,1), chair’] Class Attributes: [‘(0,1), pot’, ‘(1,1), pot’, ‘(1,1), chair’, ‘(2,1), person’, ‘(2,1), furnishing’, ‘(2,1), chair’] Explanation: I am in floor 3, because I see pot at left top, chair at center, furnishing at right center and chair at right center.
	Classification: Floor 5 $E_{M,5}$: [‘(2,1), chair’, ‘(2,1), furnishing’, ‘(2,2), chair’] Class Attributes: [‘(1,1), chair’, ‘(1,1), furnishing’, ‘(1,1), table’, ‘(1,2), chair’, ‘(2,1), chair’, ‘(2,2), furnishing’, ‘(2,1), furnishing’, ‘(2,2), chair’] Explanation: I am in floor 5, because I see chair at right center, chair at right center and furnishing at right bottom.
	Classification: Floor 6 $E_{M,6}$: [‘(2,1), pot’, ‘(2,1), sofa’] Class Attributes: [‘(1,1), person’, ‘(2,1), furnishing’, ‘(2,1), chair’, ‘(2,1), pot’, ‘(2,1), sofa’] Explanation: I am in floor 6, because I see pot at right center and sofa at right center.

Table 1: Explanations for three floors of our building.

GridLabel(gId_1), ... and Label(o_n) at GridLabel(gId_n).

Robot Explanation Demonstration

We tested our algorithm’s ability to explain the classification for each of the floors in our building. The intermediate and final outputs are provided for three example floors in Table 1. For the first image representing Floor 3, the multi-object detector finds six explainable features that have more than $t = 0.5$ of their area included in the importance mask – E_M = [‘(0,1), pot’, ‘(1,1), pot’, ‘(1,1), chair’, ‘(2,1), person’, ‘(2,1), furnishing’, ‘(2,1), chair’]. However, when compared to the class attributes, only four of the six features are included in $E_{M,3}$, as shown in Table 1. The natural language explanation for the Floor 3 image is: *‘I am in floor 3, because I see pot at left top, chair at center, furnishing at right center and chair at right center.’*

5 Practical Implementation Results and Discussion

In implementing the explanation algorithm, we observed several practical requirements and made several design decisions that affect applicability to new domains.

First, it is important to choose a secondary classifier that can actually label the features in the domain and ideally produce unique attributes per class. Two of our floors (2 and 7) have very few features in our training images. In fact, the classifier was unable to find any features in any training images for Floor 2. Since no other class attribute set was empty, the lack of explainable features was considered unique to that class. In our testing, Yolo9000 could not detect any objects for two images from 'Floor 7' either, since the image contained nothing other than a brown wall (sometimes there was a plant in view). The inability to distinguish between Floors 2 and 7 using those images were the only two errors in our classifier. Training Yolo9000 to detect walls in general (it labeled walls as 'electric refrigerator' in some images as shown in Figure 2(c) and specifically the difference between brown and white walls would have improved this result. However, overall the algorithm was very robust to our classifier and threshold choices.

Next, we made a decision to include only the explainable features in the image that also occurred within the class attributes. While taking an intersection to generate $E_{M,c}$ provides a good explanation in our domain, we also considered other ways of determining which features to include. In one alternative, we included all objects from E_M. As a result, for example, for Floor 3, the algorithm explains seven features ['(0,1), pot', '(1,1), pot', '(1,1), chair', '(1,2), seat', '(2,1), person', '(2,1), furnishing', '(2,1), chair'] rather than four. In practice, we found that removing features that were not important for the class (e.g., seat) reduced some of the confusion about whether the feature was important for the image or for the classification. Another alternative was to use the explainable features that represented the difference between $E_{M,c}$ and $E_{c'}$ $\forall\, c' \in C$ & $c' \neq c$. This algorithm explains three features ['(0,1), pot', '(1,1), pot', '(2,1), person'] which uniquely differentiate it from all other classes. In practice, this selection algorithm often produced too few and/or empty feature lists because all the classes contain a similar set of objects but in different positions. Choosing to explain $E_{M,c}$ was a good balance between including enough features to be useful and limit the inclusion of the features that were not important.

Finally, our implementation built on top of the extensive prior work done in finding important regions of images. To extend this work to classifiers operating on non-image data, we suggest using other existing importance functions such as PCA or a modified version of the deep visualization technique. We used an image-based multi-object detector, but any classifier over subsets of features would be applicable. Even the discretization of our image for an explanation can be adapted to non-vision data. For example, rather than using full timestamps in temporal data, it is possible to discretize events by morning, afternoon, evening, and night.

6 Conclusion

We contribute an algorithm for automatically generating an explanation for classifiers using natural language. In order to avoid explaining all features, our algorithm first finds the important regions of the image. Then, using a secondary pre-trained classifier, it finds the explainable features that makeup only the important regions of the image and uses the labels as the natural language description. By relying on a separate multi-object detector, our algorithm can label and explain the images without requiring manual labellings. The algorithm then determines which of the image's features are common to the class and generates an explanation of only those features. Our algorithm successfully works on our mobile service robot which uses a CNN classifier. The methods we developed also provide good directions to expand its applicability to non-visual domains.

References

[1] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 582. ACM, 2018.

[2] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[3] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[4] Zvi Boger and Hugo Guterman. Knowledge extraction from artificial neural network models. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 4, pages 3030–3035. IEEE, 1997.

[5] Yuan Been Chen and Oscal TC Chen. Image segmentation method using thresholds automatically determined from picture contents. *EURASIP Journal on Image and Video Processing*, 2009(1):140492, 2009.

[6] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 303–312. ACM, 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

[9] B Karthikeyan, V Vaithiyanathan, B Venkatraman, and M Menaka. Analysis of image segmentation for radiographic images. *Indian Journal of Science and Technology*, 5(11): 3660–3664, 2012.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Frederic Lehmann. Turbo segmentation of textured images. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):16–29, 2011.

[12] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[13] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. 2016.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[15] Stephanie Rosenthal, Sai P Selvaraj, and Manuela M Veloso. Verbalization: Narration of autonomous robot experience.

[16] Ivan Sanchez, Tim Rocktaschel, Sebastian Riedel, and Sameer Singh. Towards extracting faithful and descriptive representations of latent variable models. *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*, 2015.

[17] R. Selvaraju, Das A., R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[18] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014.

[19] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996, 2014.

[20] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[21] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016.

[22] Z. Zheng, L. Zheng, and Y. Yang. A discriminatively learned CNN embedding for person re-identification. *CoRR*, abs/1611.05666, 2016.

[23] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[24] L. Zintgraf, T. Cohen, and M. Welling. A new method to visualize deep neural networks. *CoRR*, abs/1603.02518, 2016.